Лабораторна робота 1. Мікросервісна система. Створення blank solution в Visual Studio 2019. Додавання чотирьох проектів API для мікросервісів Products, Customers, Sales, Search.

Мета. Вивчити особливості мікросервісної архітектури, отримати практичні навички розробки мікросервісів на платформі .NET Core в середовищі розробки Visual Studio 2019 (мова C#).

Завдання: Розробити мікросервісну систему, що включає чотири мікросервіси: Products, Customers, Sales і Search.

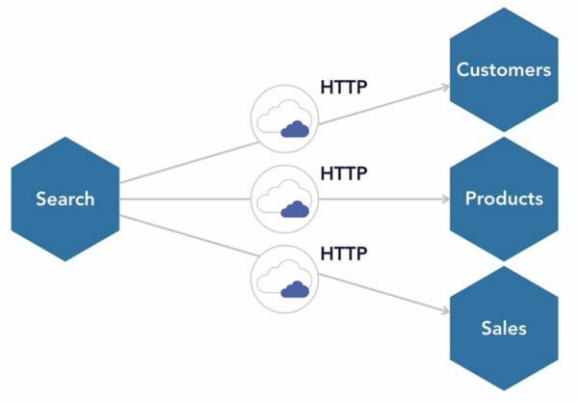

Рис. 1. Мікросервісна система з чотирьох сервісів.

Перші три мікросервіси повинні управляти власними даними. Вони повинні зберігати дані в пам'яті, хоча це швидко можна замінити на базу даних.

Мікросервіс Search повинен синхронізовано взаємодіяти з іншими мікросервісами для отримання даних від конкретного користувача. Мікросервіс Search повинен повертати об'єднані дані. Для досягнення цього ми повинні вирішити наступні завдання:

1. Ми розробимо мікросервіси використовуючи Visual Studio 2019 і .NET Core.
2. Потім ми створимо образи (image) для контейнера Docker і Docker Composer скрипт для локального відлагодження.
3. Далі ми помістимо (push) вихідний код до централізованого репозиторію, подібному Azure DevOps.
4. Далі з використанням комунікаційних магістралей (pipelines) робимо численні побудови (build) і релізи (release) щоб впевнитись, що все працює так, як потрібно, і щоб побудувати образи (image) для контейнера Docker.
5. Після цього ми заштовхнемо (push) образи Docker в Container Registry. Для прикладу, в Docker Hub і Azure Container Registry.
6. І на завершення ми розгорнемо всі контейнери в Azure Service Fabric Claster.

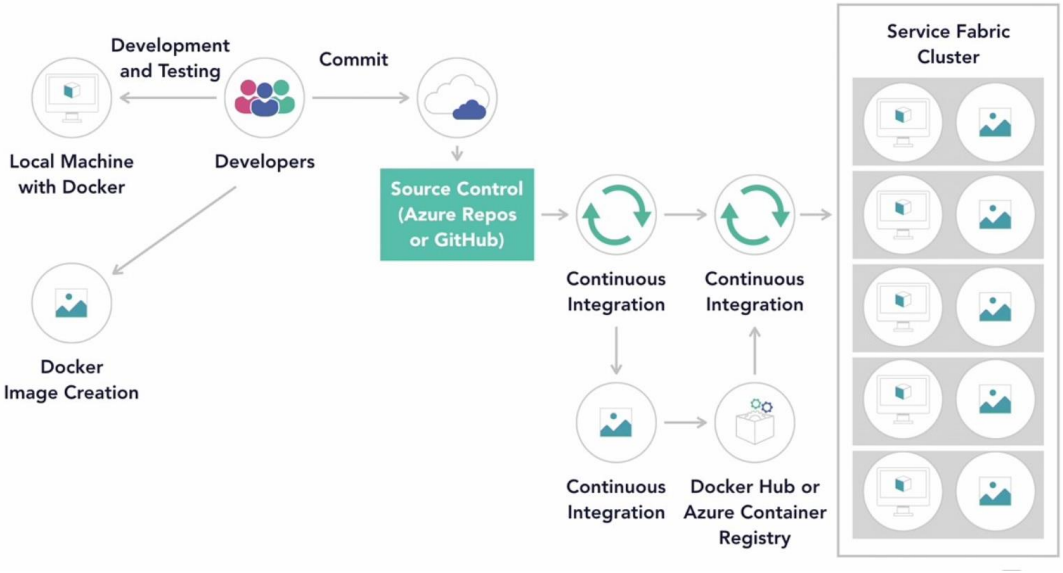

Рис. 2. Етапи побудови мікросервісної системи.

Порядок виконання.

1. Створення пустого Рішення.

Відкрийте Visual Studio 2019, виберіть File => New Project. В діалоговому вікні в рядку пошуку наберіть Blank Solution. Виберіть шаблон Blank Solution і натисніть Next.

Далі виберіть Location (наприклад D:\course), ім'я проекта (**ECommerce**) і натисніть Create.

Для кожного мікросервісу ми будемо створювати окремий проект.

2. Додавання проекту для мікросервіса Products.

Натисніть праву кнопку на Solution “ECommerce”, виберіть Add, New Project, в діалоговому вікні виберіть шаблон ASP.NET Core Web Application і натисніть Next.

Введіть ім'я проекту **ECommerce.Api.Products** і натисніть Create.

В наступному діалозі треба вибрати шаблон для проекту. Виберіть шаблон API і зніміть галочку біля опції Configure for HTTPS. Натисніть кнопку Create і почекайте поки проект створиться.

3. Додавання проекту для мікросервіса Customers.

Рухаємось далі. Повторіть все з попереднього пункту для створення проекту для мікросервісу Customers, давши ім'я проекту **ECommerce.Api.Customers**.

4. Додавання проекту для мікросервіса Orders.

Рухаємось далі. Повторіть все з пункту 2 для створення проекту для мікросервісу Orders, давши ім'я проекту **ECommerce.Api.Orders**.

5. Додавання проекту для мікросервіса Search.

І на завершення. Повторіть все з пункту 2 для створення проекту для мікросервісу Search, давши ім'я проекту **ECommerce.Api.Search**.

6. Видалення зайвих файлів.

Відкрийте проект Products в Solution Explorer. Розгорніть папку Controllers. Видаліть файли WeatherForecast.cs і WeatherForecastController.cs.

Видаліть також файли WeatherForecast.cs і WeatherForecastController.cs з проектів Customers, Orders і Search.

Після цього закриваємо всі документи, вибравши меню Window => Close all Documents.

7. Додавання NuGet Packages до проекту Products.

Перейдемо до реалізації мікросервісу Products. Відкрийте проект Products в Solution Explorer. Натисніть праву кнопку на Dependencies, виберіть Manage NuGet Packages. Натисніть Browse. Додамо три пакети NuGet до цього проекту.

Першим додамо пакет Microsoft Entity Framework Core, який будемо використовувати для доступу до даних. Наберіть в пошуковому рядку Microsoft.EntityFrameworkCore. Серед знайдених виберіть цей пакет. Далі натисніть кнопку Install і в вікні, що з'явилось, натисніть OK. Натисніть I Accept в вікні підтвердження ліцензії.

Далі шукаємо пакет Microsoft.EntityFrameworkCore.InMemory і приєднуємо його до проекту.

І останнім ми приєднаємо пакет AutoMapper, який будемо використовувати для об'єктно-реляційного відображення між DB Context і Models, які ми побудуємо. В пошуковому рядку набираємо Automapper.Extentions.. Буде знайдено пакет AutoMapper.Extentions.Microsoft.DependencyInjection. Інсталюйте цей пакет.

Далі натисніть праву кнопку на проекті ECommerce.Api.Products і виберіть Rebuild. У нас ще нема коду, але ми готові додати реалізацію бізнес-логіки до проекту Products.

ВИКОРИСТАНІ ДЖЕРЕЛА

Lynda - Azure Microservices with .NET Core for Developers (2020). – URL: https://www.lynda.com/Azure-tutorials/Azure-Microservices-NET-Core-Developers/2825264-2.html